504-REN-UI-TP

MISSION OPERATIONS AND DATA SYSTEMS DIRECTORATE

Renaissance User Interface Transition Plan

Original Draft

August 1994

National Aeronautics and Space Administration

Goddard Space Flight Center
Greenbelt, Maryland

Renaissance User Interface Transition Plan

Original Version

August 29, 1994

Approved By:

Gary Meyers — Date
Renaissance Team, Code 504

Goddard Space Flight Center
Greenbelt, Maryland

File ID — Document No.

Section 1. Introduction

Section 2. RENAISSANCE PROJECT SCHEDULES

Section 3. USER SERVICES TRANSITION PLAN

Section 4. BUILDING BLOCK TRANSITION PLANS

Section 1. Introduction

This section provides an introduction to the *Renaissance User Interface Transition Plan* by listing the document's purpose, scope and organization, and by detailing the set of relevant documents and terms useful to the reader.

1.1 Purpose and Scope

The *Renaissance User Interface Transition Plan* describes the overall plan for transitioning legacy user interfaces developed within the Mission Operations and Data Systems Directorate (MO&DSD) into the form required for the Renaissance project. The document also describes the long-term plan for the evolution of user interface technology once the initial integration of legacy software is complete. This document is intended for managers and system developers who need to plan and implement changes in existing user interface software building blocks to allow their incorporation into the Renaissance user environment.

This document is maintained by the User Services Working Group which reports to the MO&DSD Renaissance Team. The working group will update the document to reflect any changes made to the schedule or approach for the evolution of Renaissance user services building blocks.

1.2 Document Organization

This document is divided into four sections. The first section provides this background information on the purpose and organization of the document. Section 2 describes the overall Renaissance project schedule and any existing Renaissance commitments for support of upcoming spacecraft missions. Section 3 describes the overall transition approach for the user services software within the Renaissance project. Finally, Section 4 defines the steps that need to be taken to update each candidate software building block within MO&DSD to operate in the Renaissance user environment.

1.3 Related Documents

For additional information about Renaissance and the user services working group, consult the following documents:

1. *Open Software Foundation (OSF)/Motif Style Guide*

2. *Renaissance User Interface Implementation Guide*

3. *Renaissance User Services Building Block Specifications*

1.4 Terminology

The following terms are defined in this section and appear throughout the document:

Application: A *software element* external to the sphere of the USWG for which the USWG must provide a *user interface.* Although many *applications* will fall into the sphere of the Applications Services Working Group, this definition permits elements from the data services, simulation, and spacecraft communication working groups to be considered *applications.* This term is most commonly used in this document to refer to the non-user interface related portion of a software function.

Building block: A loose term for generic or tailored *subsystem, element* or *subelement.*

Component: An element within a graphical user interface (GUI) that is identifiable to the user as a distinct entity. Examples of *components* include push buttons, menu bars, and text fields.

Generic: A generic *system, subsystem, element* , or *subelement* is one that is always used with no modifications from mission to mission. Mission configurability may be achieved through data parameters or through tailoring .

Guideline: A characteristic of a software *building block* that can be evaluated to determine the degree of applicability that *building block* possesses for use within the Renaissance environment. *Guidelines* will be used both to steer the development of new *building blocks* and to evaluate the legacy *building blocks* available within MO&DSD to determine which should become the baseline set of Renaissance *building blocks.*

Mission-Specific: A mission-specific *system, subsystem, element* , or *subelement* is one that is developed and managed for a specific mission.

Software Backplane: The set of Renaissance-standard APIs for providing communications between software elements. Conceptually, elements "plug into" the standard interfaces of the backplane (in analogy to a hardware backplane).

Software Element: An independently executing software unit that communicates with other software elements solely through the standard interfaces defined by the Renaissance software backplane. A software element is constructed from subelements that may be linked together or may be separate tasks or processes. Subelements within a software element may communicate among themselves by means other than the backplane.

Software Subelement: A configuration-controlled component of one or more software elements. Subelements may themselves have structure, but the subelement level is the lowest level of concern for configuration of a system for a particular mission.

Standard: A set of principles, conventions, and/or programming interfaces established by either the computer industry or the Renaissance team to govern the development of software. The X Window System, OSF/Motif, and Common Desktop Environment (CDE) are all examples of industry standards.

Subsystem: A standard decomposition of a *system*, encompassing those *software elements* necessary to carry out a specific high-level functional area. For example, Spacecraft Operations Monitoring and Mission Planning might be *subsystems* (these examples are for illustrative purposes only, and do not imply a definitive decomposition of Renaissance systems).

System: A complete operational environment with clear boundaries and well-defined external interfaces crossing these boundaries, such as the Mission Operations Center (MOC), Science Operations Center (ASC) and spacecraft Integration and Test (I&T) systems.

Tailored: A tailored *system*, *subsystem*, *element* or *subelement* is one with some mission-specific code added to a generic part in some standard way (e.g., linked in via well-defined hooks).

User interface: A *software element* that falls within the sphere of the USWG. Thus, from the point-of-view of the USWG, all *software elements* are either *applications* or *user interfaces*.

Widget: A *software subelement* that conforms to the X toolkit and Motif widget library standards for implementing user interface components. Examples of widgets are XmPushButton and XmTextField. Custom widgets created for the National Aeronautics and Space Administration (NASA) environment are also discussed in this document.

Section 2. RENAISSANCE PROJECT SCHEDULES

The characteristics and schedule for each of the missions to be supported by the Renaissance project are listed in the sections below. This information is used to set the stage for the overall user services transition plan presented in Section 3.

2.1 ACE Schedule

The Advanced Composition Explorer (ACE) mission's primary science objective is to observe particles of solar, interplanetary, interstellar, and galactic origins , spanning the energy range from that of the solar wind (1 keV/nucleon) to galactic cosmic ray energies (several hundred MeV/nucleon). ACE will also monitor the solar wind and provide real-time data to scientists. There will be nine independent science instruments onboard the spacecraft.

ACE is scheduled for launch from the Cape Canaveral Air Force Station (CCAFS) aboard a Delta II 7920 Expendable Launch Vehicle (ELV) in August 1997. The ACE spacecraft will be constructed by the Johns Hopkins University (JHU) Applied Physics Laboratory (APL), who is also under contract to GSFC Code 400 for the integration and test (I&T) of the ACE spacecraft and science instruments.

ACE breaks from the traditional NASA approach where ground systems for mission operations, science operations, and spacecraft I&T are usually developed independently of each other. Instead, the ACE project will use the same open systems, distributed architecture for all three of these ground system environments. The Transportable Payload Operations Control Center (TPOCC) real-time control center software has already been selected for use in all three areas, and other Renaissance software building blocks are also expected to be shared among these elements. The current ACE implementation plan calls for releases of this shared "ACE generic" software as early as the end of this calendar year to begin support of the I&T environment. This requirement to support ongoing spacecraft testing from July 1995 until launch is a critical constraint on the flexibility afforded to the Renaissance project in modifying existing building blocks for a new Renaissance software architecture.

2.2 SMEX-4 SCHEDULE

The SMEX-4 project is the fourth spacecraft to be launched in the series of Small Explorer (SMEX) missions. The first SMEX mission was SAMPEX, which was launched in July 1992. The next two missions, FAST and SWAS, are both scheduled to be launched in 1995. All SMEX missions are low-cost scientific projects with the spacecraft construction and integration and test performed at GSFC. These projects emphasize mission-to-mission reuse of both spacecraft and ground system hardware and software to realize their cost goals.

The science mission for SMEX-4 has not yet been selected, though it has been narrowed down to either a TRACE/POEMS combination mission (designated TRAPEM) or WIRE. The characteristics of the mission will differ depending on the science objective selected (e.g. TRAPEM downlinks 450 Mbytes of science a day, WIRE only 95 Mbytes). It is known that the maximum rate of spacecraft downlink during a tape recorder dump will exceed ACE's 76 Kbps by at least an order of magnitude (1.8 Mbps for WIRE and 2.25 Mbps for TRAPEM). Thus, the Renaissance

control center built for SMEX-4 will need to have much greater performance capacity than the first Renaissance control center built for ACE.

2.3 FUSE Schedule

The Far Ultraviolet Spectrograph Experiment (FUSE) project is currently scheduled for spacecraft launch near the end of the year 2000. This project is currently in the very early stages of development, with the start of Phase C/D scheduled for October 1995. This schedule affords about a three year gap between ACE and FUSE. For this
reason the Renaissance project expects to have completed the majority of the transition efforts by the time FUSE launches.

2.4 Future Spacecraft

Although no spacecraft missions have been assigned to Renaissance beyond FUSE, it is anticipated that most or all of the upcoming missions supported by MO&DSD will be implemented as Renaissance systems. One near term possibility for Renaissance is the LANDSAT-7 project, though no final decision has been made on its disposition as a Renaissance project.

Section 3. USER SERVICES TRANSITION PLAN

This section of the document describes the overall transition approach for applying existing MO&DSD user interface technology to the initial set of Renaissance projects that were described in Section 2. The approach evolved from the need to impose standards and guidelines for Renaissance user interface building blocks in order to meet the Renaissance user interface goals documented in the *Renaissance User Interface Implementation Guide*. The following table summarizes the planned phase-in of the standards and guidelines listed in the *Renaissance User Interface Implementation Guide* across upcoming missions to be supported by the Renaissance architecture. The subsections that follow describe the activities that will be undertaken in support of each upcoming mission to make this phase-in possible.

STANDARDS & GUIDELINES:	**ACE**	**SMEX-4**	**FUSE**	**Future Spacecraft**
X11	R5	R5	R6	R6
Motif	1.2.3	1.2.3	2.x	2.x
Common Desktop Env	basic integration	partial integration	full integration	full integration
Ren Style Principles	partial	partial	full	full
Language	C or C++	C or C++	New BB in C++	New BB in C++
POSIX	identify	isolate	compliant	compliant
Ren Develop Stds	TBD	TBD	TBD	TBD
Import/Export	within BB	within BB	across BB	across BB
Scripting	individual methods	individual methods	partially integrated	common method
Customizable by user	partial	partial	full	full
Decoupled from app	change in key areas	change in key areas	decoupled	decoupled
Layout/code separate?	multiple layouts	multiple layouts	mostly standard	standard layout
Custom widgets	establish library	establish library	consistent library	consistent library
Platforms	HP	HP	HP, Sun?	All ref platforms
COTS usage	current COTS	current COTS	compliant COTS	compliant COTS
Ren Interfaces	partial	partial	full	full
Documentation	complete	complete	standard format	standard format
Maturity	some early releases	some early releases	more mature	fully mature
On-line Help	initial	initial	full	full
Performance	satisfies reqmts	satisfies reqmnts	abort & feedback	abort & feedback
Portability	tried	tried	< 1 month	< 1 month
Support Availability	basic	basic	enhanced	routine
Verbatim reuse	yes	yes	yes	yes

Renaissance Missions versus Standards and Guidelines

3.1 ACE Goals

Since ACE is the first mission that will be supported by Renaissance, the USWG realizes that goals for the ACE user environment should be limited to the highest priority activities. The transition goals chosen for ACE are those that the working group felt would maximize the consistency of the user environment at the least cost. The selected goals include:

- supporting basic integration of all building blocks with the Common Desktop Environment (CDE) and providing online help, probably in the CDE format
- implementing the top priority user interface style principles selected by Renaissance
- decoupling the user interface software from underlying applications in one or two key areas
- establishing and populating the Renaissance widget library
- beginning the transition to the Renaissance software backplane in key areas
- creating a common event format

Each of the ACE goals is described briefly in a paragraph below. Also note that the USWG realizes that for ACE there may be some overlap in capabilities between user interface building blocks. Over time, Renaissance will strive to eliminate this redundancy and provide only a single building block that supports each functional capability. However, the schedule pressures from the ACE project make this a goal that will not be satisfied in the near term.

The first ACE goal is basic integration of all building blocks with the Common Desktop Environment (CDE). CDE is a project led by the UNIX workstation vendors to create a standard user environment such as those existing for Macintosh users across all Macintosh PCs or with the Windows environment for PC users. Announced in late 1993 by the Common Open Software Environment (COSE) consortium of vendors, CDE builds on the most prevalent user interface technology available for UNIX systems at this time: X Windows, OSF/Motif, Hewlett Packard's Visual User Environment (VUE) and Sun's ToolTalk messaging protocol. To achieve basic CDE compliance, all applications must comply with the Motif look-and-feel and must support their installation into the CDE desktop environment. Initial support of the CDE help system is also a requirement of basic integration. Since almost all of the Renaissance user interface building blocks have not developed online help systems to date, the USWG is hoping to capitalize on this situation and encourage all of the projects to develop their help systems with the CDE help package.

The *Renaissance User Services Implementation Guide* will contain a detailed section on user interface style principles that will establish a Renaissance look-and-feel that is consistent with the Motif standards. For ACE, each building block will need to make the highest priority changes in this area. Examples could include shifting their time representations into a consistent ASCII string format and following a naming convention for all window titles. The style principles are currently scheduled for completion by early 1995.

Another area where the highest priority items will be tackled for ACE is the decoupling of user interface software from underlying applications. This transition needs to be addressed for three of the current user services building blocks and will have to be handled carefully for it probably

requires redesigns for two of the building blocks (GenSAA, and GTAS) and could introduce performance and compatibility issues for the third (UIX).

Establishment of a Renaissance widget library support team and population of the library with display components of general utility is a key activity for the long-term future of the Renaissance user environment. Although the library will not be entirely consistent or complete in the ACE timeframe, getting an early start on this activity is important. Rapid availability of widgets from this library to the building block projects will help the user interfaces achieve the consistency goals. A good example of this is the benefit that would be derived from a time input widget that could be used by all building blocks to alleviate the current problem of a multiplicity of time representations.

An additional area where the highest priority items will be tackled for ACE is the transition to the Renaissance software backplane. The highest priority software backplane item for the USWG is establishment of a common system event message representation, logging, and distribution scheme. A common events approach will allow the user to browse through logs showing all of the activity within the ACE Mission Operations Center (MOC) regardless of the development legacy of the software that generated the message. This item is absolutely critical so as not to impede the smooth flow of operations activities in the ACE MOC. Initial work to define and document such a common event scheme is currently underway between the user and data services working groups.

3.2 SMEX-4 GOALS

The SMEX-4 spacecraft launch is currently scheduled to occur in March 1988, just seven months after the ACE launch. Because of the roughly parallel development schedules for the ACE and SMEX-4 projects, the USWG does not anticipate any substantial differences between the Renaissance building blocks to support the two missions. Thus, only one improvement has been targeted for the SMEX-4 timeframe, namely achieving greater software portability by isolating all POSIX system calls in separate libraries. If in the next three years the ACE and SMEX-4 schedules diverge, additional SMEX-4 transition goals will be set.

3.3 FUSE Goals

As the third Renaissance mission with launch over 5 years away, FUSE represents the first chance to present a truly unified user environment. The USWG has thus set goals for the FUSE user environment that reflect a relatively mature set of building blocks with most transition activities completed. The selected goals include:

- full integration of all building blocks with the Common Desktop Environment (CDE) and comprehensive online help
- full implementation of user interface style principles selected by Renaissance
- POSIX compliance
- Motif-style data transfer between user interface building blocks
- increased support of user customization
- decoupling of the user interface software from underlying applications in all areas
- complete implementation of the Renaissance software backplane

- consistent and complete sets of Renaissance widgets and documentation

Since each of the USWG standards and guidelines are fully documented in the *Renaissance User Services Implementation Guide*,, detailed information on these FUSE goals is not repeated here.

3.4 Long-Term Goals

Finally, the USWG has identified some long-term goals that may still be too ambitious to achieve completely in time for support of the FUSE system. However, it is anticipated that substantial progress will be achieved for FUSE in these areas even if these guidelines are not completely satisfied. These long-term goals are as follows:

- a common capability for scripting and automating user activities that works across multiple building blocks

- a standard representation for all displays so they can be freely interchanged among building blocks

- proven portability between POSIX platforms including routine support for Renaissance software on several Reference platforms

- well established support offices to assist system integrators using each building block

Finally, the USWG has set a long-term goal of eliminating any overlap in capabilities between user interface building blocks. Hopefully, by the turn of the century the Renaissance user interface software will be a consistent suite of cooperating user interface building blocks each contributing a set of unique functional capabilities.

Section 4. BUILDING BLOCK TRANSITION PLANS

A comparison of the current state of each Renaissance user services building block with the USWG's standards and guidelines follows in the sections below. The first section describes the current status and action plan for custom
widgets developed within Code 500 that can be shared among projects. The descriptions that follow each analyze one of the user interface software elements that will be used in initial Renaissance systems.

4.1 RENAISSANCE Widget Library

The USWG has identified user interface consistency as their principle goal so that the end user of a Renaissance system is presented with an integrated user environment. Although all user interface software elements are currently layered over the Motif widget library and conform to Motif style principles, there is still much room for inconsistency when user interface components are developed that fall outside of the Motif sphere. A perfect example of such a case is provided by the requirement most of these building blocks share to allow user input of time information (e.g. start and stop time of data replay). At least four different visual representations of time currently exist among the user interface building blocks under consideration. Worse yet, two inconsistent custom widgets have been created to make input of time information easier within their particular user domain. At the least, this type of inconsistency is highly annoying to a Renaissance user; at the worst it could place a spacecraft in danger if the user is unable to enter critical information because of confusion in input syntax.

Realizing that such inconsistencies exist, the USWG has begun work to establish style principles that will govern the format of information, including times. In addition, the USWG has identified the need for a common set of user interface components that are specific to the NASA environment. Many components of this type have already been written for the various projects within MO&DSD. These software subelements have been implemented according to the X toolkit industry standard that provides a stylized approach for developing user interface objects known as widgets. The USWG is proposing that these widgets be consolidated into a single library maintained by the Renaissance project. Thus, all Renaissance user interfaces could share a common time input widget to alleviate the problem described above.

The following subsections describe the approach for transitioning existing MO&DSD widgets into the realm of the Renaissance widget library. The first subsection compares the existing legacy widgets with the user services standards and guidelines defined in the Renaissance User Interface Implementation Guide. The remaining subsection describes the actions to be taken to create and maintain this library and the schedule for completing these activities.

4.1.1 Mapping to Standards and Guidelines

This section presents the results of the comparison of the current Renaissance widget implementations with the standards and guidelines presented in the Renaissance User Interface Implementation Guide. The results are divided up into the four categories of standards and guidelines defined in that document.

4.1.1.1 User Interface Standards

The status of Renaissance widget compliance with user interface standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
X11	R5	
Motif	1.2.2	
Common Desktop Env	n/a	CDE integration is at the user interface process level
Ren Style Principles	yes, so far	Strive for compliance with Motif Style Guide

Current sources of custom widgets within Code 500 are TPOCC (around 20 widgets, mostly for real-time data monitoring), GenSAA (with around five widgets representing objects on a spacecraft schematic display), and MOPSS, PICS, and QAWS, which each contain 1 to 4 general utility widgets for time input and other miscellaneous functions. All of these projects are currently based on X11R5 and Motif 1.2. Thus, no problems are anticipated with the compatibility of the contributed widgets with these commercial libraries. In addition, CDE compliance is a non-issue since it applies to complete user interface applications only. All of these projects have always worked toward OSF/Motif Style Guide compliance. It is anticipated that the widgets will thus have a high degree of compatibility with the Renaissance style principles that will be based on this style guide.

In summary, there is no significant transition for the Renaissance widget candidates to meet the user interface standards. Some changes to conform to Renaissance style principles may be required but are not anticipated to be major in nature.

4.4.1.2 Renaissance Standards

The status of Renaissance widget compliance with Renaissance standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
Language	C	C is required to write Motif widgets
POSIX	unsure	Most API calls are to X and Motif libraries
Ren Develop Stds	TBD	

All Motif widget development must be performed in C according to the conventions of the X toolkit. Thus, the candidate widgets will all be similar in implementation language and coding style. POSIX system calls are not usually required to achieve the functionality of a widget. POSIX compliance should be verified, however, as these widgets are received for insertion into the library.

In summary, the only possible change anticipated for Renaissance widgets to conform to the current Renaissance standards is in the area of POSIX compliance. In addition, future Renaissance requirements to follow development standards could necessitate some additional changes to the widget development and maintenance approaches.

4.4.1.3 User Interface Guidelines

The status of Renaissance widget compliance with user interface guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Import/Export	partial	Motif drag-and-drop implemented for some
Scripting	n/a	
Customizable by user	yes	Through UIL and resource files
Decoupled from app	n/a	
Layout/code separate?	n/a	
Custom widgets	over 30	Widgets will be received by outside contribution

Many of the guidelines listed in this area do not apply to individual widgets. Thus, scripting, decoupling, and separation of layout and code are not evaluated for the candidate widgets. Some of the widgets do support the Motif
drag-and-drop protocol, a capability that will enhance data import and export within applications that use these display components. Although many of the widgets support drag-and-drop through inheritance from Motif text and label classes, the TPOCC mnemonic browser widget actually has a built-in capability to transfer the entire complement of information on a telemetry point to an interested Motif application. Some of the widgets could be enhanced to respond to this type of data transfer, but that would be a long-term goal.

Customization is a capability that must be built in to any X toolkit widget through the standard resource mechanism. This technique allows the widget's attributes to be defined at compile time, during software configuration in a UIL or X resource file, or an run-time through command line options.

Each of the widgets from the sources listed in 4.4.1.1 above will need to be contributed to the widget library. Section 3 of the Renaissance User Services Implementation Guide lists the complete requirements for such a software transfer to the Renaissance widget library support team. Some of the products required to transfer a widget into the library are listed here:

- source code
- build dependencies
- UNIX man page documentation for the widget and its public procedures
- test suite (either sample UIL definitions or a test program)

In summary, the only real effort associated with this category will be the work required of the library support team in accepting and integrating new widgets. This team will be responsible for ensuring the quality and consistency of the
widget library and adding these display objects into any Renaissance tools that manipulate widgets such as the UIL compiler and a drawing tool (Builder Xcessory perhaps).

4.4.1.4 Renaissance Guidelines

The status of Renaissance widget compliance with Renaissance guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Platforms	varies	See information on contributing projects
COTS Usage	none	
Ren Interfaces	TBD	
Documentation	partial	TPOCC and PACOR II widgets are documented
Maturity	varies	See information on contributing projects
On-line Help	no	
Performance	yes	
Portability	varies	See information on contributing projects
Support Availability	no	Will be established along with the library
Verbatim reuse	yes	

The candidate widgets do not all share the same level of compliance with the Renaissance guidelines established by the USWG. Information on supported platforms, software maturity, and portability will vary according to the contributing project. Similarly, the TPOCC and PACOR II widgets have UNIX man pages written for them, the other widgets to be contributed do not. No widgets are dependent on any COTS product other than X Windows and Motif themselves. None are currently linked to an online help system. No problems with their performance or reusability is anticipated. Finally, support for these widgets will become available as soon as the widget library maintenance team is established.

In summary, the only real effort associated with this category will once again be linked to the establishment of the library itself. The documentation and portability of the various widgets will need to be reviewed for acceptability as they are contributed. In a similar vein the library support team might want to publish the manual pages in a format so that they are accessible as part of the Renaissance help system.

4.1.2 Action Plan

The action plan for the Renaissance widget library is perhaps the most difficult to ascertain since no final decision has been made on how the funding to maintain the library will be obtained. The initial assumptions of the USWG is that money from each successive Renaissance mission would be contributed to the library maintenance team to fund their ongoing activities.

Another decision needs to be made concerning the organization that will oversee these activities. The most cost effective solution is probably to grant the Mission Operations Division, Code 510, with the maintenance of this library since they have extensive experience in widget library maintenance as part of the TPOCC project. Maintenance costs should be substantially reduced, maybe halved, if the work falls within this Center of Expertise (COE).

Finally a schedule should be established for the establishment of this library. There is an obvious benefit to user interface development efforts if a wide range of widgets are available as soon as possible. However, this benefit
should be weighed against the fact that Renaissance user interface style principles and complete widget contribution procedures have not yet been established. Once in place, these products would allow the widgets to be brought up to full Renaissance standards by the original implementation team and then added to the library. This approach is probably more cost effective than requiring the changes be made by the library maintenance team that did not author the original software. With this in mind, a date of March 1995 could be set for establishment of the style principles and contribution procedures. Completion of these items would allow all widget contributions to be completed in calendar year 1995.

4.2 Transportable Payload Operations control Center (TPOCC) Display Subsystem

The Transportable Payload Operations Control Center (TPOCC) display subsystem is a collection of software processes that support real-time spacecraft command and control. For spacecraft health and safety monitoring, this software utilizes a library of real-time display widgets that support display of data as text, gauges, or plots. These widgets are grouped into a series of display windows for a particular project that are stored in Motif's User Interface Language (UIL). These multiple displays can all be accessed during a spacecraft contact in a multi-window environment.

TPOCC's support of ground system and spacecraft command and control is through the TPOCC System Test and Operations Language (TSTOL). The TPOCC display subsystem provides a standard environment for working with this language that allows all operator input (manual or from scripts called "procedures") to be logged and reviewed. An enhanced commanding environment, known as the command panel, is also included with this display system The command panel allows for rapid execution and monitoring of a pre-defined sequence of TSTOL procedures known as a pass plan.

The following subsections describe the approach for transitioning the TPOCC display subsystem for use with Renaissance. The first subsection compares the existing legacy building block with the user services standards and guidelines defined in the *Renaissance User Interface Implementation Guide.* The remaining subsections describe the actions to be taken to update the TPOCC display subsystem and the schedule for completing these activities.

4.2.1 Mapping to Standards and Guidelines

This section presents the results of the comparison of the current TPOCC display subsystem implementation with the standards and guidelines presented in the *Renaissance User Interface Implementation Guide.* The results are divided up into the four categories of standards and guidelines defined in that document.

4.2.1.1 User Interface Standards

The status of TPOCC display subsystem compliance with user interface standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
X11	R5	Since Release 9
Motif	1.2.2	Since Release 9; considering upgrade to 1.2.4
Common Desktop Env	partial: VUE	Integrated with HP VUE since Release 6 (a principle CDE component)
Ren Style Principles	yes, so far	TPOCC User Interface Subgroup reviews for Motif Style compliance

The TPOCC display subsystem is a mature product that has been based on the X Window System and OSF/Motif technologies since 1990. Release 9 of TPOCC used X11R5 and Motif 1.2; the most recent TPOCC release available in July 1994 is Release 11, which utilizes these same X and Motif versions. TPOCC projects have also been using the Hewlett Packard (HP) Visual User Environment (VUE) desktop environment since late 1991. HP VUE is a critical piece of the Common Desktop Environment (CDE) technology which will be used for Renaissance. The TPOCC display software will certainly require significant changes for a move to full CDE compliance; however, the fact that TPOCC already utilizes the HP VUE environment should ease

that transition somewhat. Finally, TPOCC has always had a user interface group that reviews new user interface elements for consistency with the *OSF/Motif Style Guide* and other TPOCC user interface elements. Thus, TPOCC should have a high degree of compliance with the Renaissance style principles that are being based on this document.

In summary, TPOCC's only significant transition to meet the user interface standards requirements is to achieve CDE compliance. Some changes to conform to Renaissance style principles may be required but are not anticipated to be major in nature.

4.2.1.2 Renaissance Standards

The status of TPOCC display subsystem compliance with Renaissance standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
Language	C	Studying transition to C++ for new TPOCC software elements
POSIX	partial	Have isolated platform-specific system calls in separate libraries
Ren Develop Stds	TBD	Development standards are still TBD

All TPOCC development is currently in C, though initial studies on migration to C++ have been undertaken. No firm plans or timetable for such a migration exist at this time. TPOCC is currently working to improve their POSIX compliance. The project has recently implemented POSIX signals and has prototyped the conversion to POSIX shared memory access. Most operating system dependencies within TPOCC are isolated in independent software libraries which also aids in the portability of TPOCC's system calls.

In summary, the only change currently anticipated for TPOCC to conform to the current Renaissance standards is a continuation of the effort to reach POSIX compliance. However, future Renaissance requirements to follow development standards or utilize object-oriented techniques could necessitate further changes to the TPOCC development approach.

4.2.1.3 User Interface Guidelines

The status of TPOCC display subsystem compliance with user interface guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Import/Export	yes	Library of routines to support Motif drag and drop
Scripting	TSTOL	Complete procedural language
Customizable by user	partial	Completely customizable by project; partially by user
Decoupled from app	yes	Data source objects allows widgets to be animated from any source
Layout/code separate?	yes, UIL	Have extended UIL compiler to include TPOCC widgets
Custom widgets	about 20	Widget set will form foundation of Renaissance widget library

TPOCC has recently implemented a library to facilitate user transfer of data between applications through the Motif standard drag-and-drop interface. This drag-and-drop interface has been incorporated into several software processes within the TPOCC display subsystem. TPOCC supports scripting of user activity with the TPOCC System Test and Operations Language, an outgrowth of the STOL capability used successfully within the Mission Operations Division to support dozens of spacecraft over the last 25 years. TPOCC currently allows extensive customization by system integrators through the X Windows app-defaults file and UIL display

definitions. Customization by the user is only possible in the form of creating new display page definitions with either the Builder Xcessory COTS layout tool or TPOCC's own page editor.

The TPOCC user interface software is typically isolated from the underlying applications through the data server interface. The data server provides a stream of real-time data to interested clients (like the display subsystem)
running on remote hosts. Furthermore, TPOCC has decoupled the real-time display widgets in the TPOCC widget library from this interface by a move to data source objects about a year ago. This object-oriented technique allow TPOCC's widgets to be animated based on data received by any mechanism for which a subclass of the data source object has been created. This mechanism could be the data server protocol, a file interface, a pipe, or by direct calls from an application. TPOCC does maintain all layout information in the UIL language. These layouts include both the system integrator's definition of the run-time interface and any user display definitions. Finally, TPOCC has the largest collection of custom widgets within the MO&DSD, numbering around twenty. For
this reason, it is likely that the TPOCC widget library will serve as the basis for the Renaissance widget library (see Section 4.1).

In summary, TPOCC needs only to consider improvements in the area of end user customization to meet the user interface guidelines. This effort is probably more of a long-term goal than a necessity for ACE.

4.2.1.4 Renaissance Guidelines

The status of TPOCC display subsystem compliance with Renaissance standards is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Platforms	HP, Solaris	Also VxWorks for real-time embedded OS
Cost	uses BX	Support customization of BX to streamline for end users
Ren Interfaces	TBD	
Documentation	yes	Complete set of hardcopy manuals; online widget manual pages
Maturity	Rel 11	Issue new release every 5 months
On-line Help	no	
Performance	most cases	No "work in progress" dialogs, but probably not needed
Portability	< 2 weeks	Ported to Ultrix, AIX, SGI, VMS
Support Availability	yes	SEAS task has support time budgeted in
Verbatim reuse	yes	Reused on over 10 control centers

TPOCC currently supported its software on the HP, Solaris, and VxWorks platforms. Ports to the AIX, Ultrix, SGI, and VMS platforms have all been completed in the past, typically in two weeks or less. TPOCC supports their custom widgets within the Builder Xcessory user interface design tool, allowing both system integrators and end users to create new displays. This tool supports several levels of use so that new users can see a much simpler set of display options. Changes to support the Renaissance software backplane are currently undefined and could range from minor to major depending upon the similarity with current TPOCC communication mechanisms. Also note that the project routinely updates their Implementation Guide that contains complete information for system integrators.

TPOCC has been considering an online help capability but it has not received a high enough priority to be implemented up to this time. The TPOCC display subsystem may need a few improvements in providing feedback for long operations, though the real-time processing environment for which TPOCC is designed means that very few scenarios of this type exist. TPOCC has funding for a support office and the display software meets all verbatim reuse

requirements listed in the implementation guide, as nine different control centers have used the TPOCC display subsystem without modification by the mission.

In summary, TPOCC will require several changes to meet the Renaissance guidelines that have been established. The addition of an online help capability and changes to support Renaissance interfaces may be major changes.
An effort to enhance the performance feedback from the system is anticipated to be minor in nature.

4.2.2 Action Plan

The TPOCC project has an existing mechanism to handle requests for updates to TPOCC technology. This Internal Configuration Change Request (ICCR) mechanism allows any user of TPOCC technology to request a change and receive back the cost impact, a determination of whether or not this has been accepted as a new generic software capability, and the schedule for implementation. This process is fully documented in the *TPOCC Project Support Plan*. All changes to the TPOCC display subsystem identified in this transition plan will be turned into TPOCC ICCRs. Each ICCR will be costed and resources to implement the change will be allocated if the ICCR is approved. The changes will be undertaken according to the schedule listed below.

The TPOCC project delivers a major software release at least every 5 months. Releases do occur at more frequent intervals as required to meet TPOCC project requirements. Because of the changing needs of the TPOCC projects, the release schedule is usually only determined about 6 months ahead. Thus, only a tentative schedule for future TPOCC releases can be given. This best estimate is as follows:

TPOCC Release 11	July 1994
TPOCC Release 12	November 1994
TPOCC Release 13	April 1995
TPOCC Release 14	September 1995
TPOCC Release 15	February 1996

These are the dates TPOCC software is officially delivered; in most cases, the software is available for integration by TPOCC projects two months earlier.

Capabilities for implementation have been allocated through TPOCC Release 12. Thus, Renaissance requirements for ACE would need to be contained in Release 13 or later. A first guess at the schedule for updates to the TPOCC display subsystem is given below:

TPOCC Release 13:	Basic CDE integration
TPOCC Release 14:	Online help facility POSIX compliance Updates for Renaissance Style Principles
TPOCC Release 15:	Full CDE integration

4.3 Generic Spacecraft Analyst Assistant (GenSAA)

The Generic Spacecraft Analyst Assistant (GenSAA) is a tool that facilitates the development of real-time expert systems that perform spacecraft monitoring and fault isolation functions. GenSAA consists of a workbench to allow development of new expert systems and a set of run-time software that performs the monitoring during a spacecraft contact. The software is divided into three functional areas: the data manager/interface that obtains telemetry point from the TPOCC data server, the expert system (based on the NASA CLIPS software), and the user interface. The files generated by the workbench are used as input to the run-time components and specify the specific operations to be performed for a particular mission.

The following subsections describe the approach for transitioning GenSAA for use with Renaissance. The first subsection compares the existing legacy building block with the user services standards and guidelines defined in the *Renaissance User Interface Implementation Guide.* The remaining subsections describe the actions to be taken to update GenSAA and the schedule for completing these activities.

4.3.1 Mapping to Standards and Guidelines

This section presents the results of the comparison of the current GenSAA implementation with the standards and guidelines presented in the *Renaissance User Interface Implementation Guide.* The results are divided up into the four categories of standards and guidelines defined in that document.

4.3.1.1 User Interface Standards

The status of GenSAA compliance with user interface standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
X11	R5	
Motif	1.2.2	
Common Desktop Env	no	No file manager or desktop integrated to date
Ren Style Principles	yes	Strive for compliance with Motif Style Guide

The GenSAA software is currently based on X11R5 and Motif 1.2. Thus, no problems are anticipated with these commercial libraries. However, GenSAA has not previously performed integration with a desktop manager. This lack of experience may slow the process of integrating GenSAA with CDE. GenSAA has always worked toward OSF/Motif Style Guide compliance. It is anticipated that GenSAA with thus have a high degree of compatibility with the Renaissance style principles that will be based on this style guide.

In summary, the one notable transition for GenSAA to meet the user interface standards requirements is to integrate with CDE. Some changes to conform to Renaissance style principles may be required but are not anticipated to be major in nature.

4.3.1.2 Renaissance Standards

The status of GenSAA compliance with Renaissance standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
Language	C, C++	C for CLIPS and widgets; otherwise C++
POSIX	no	No attempt to achieve compliance to date
Ren Develop Stds	TBD	

All GenSAA development is currently in C++, with C used only to interface with CLIPS and to create custom widgets. The GenSAA project had no existing requirement for POSIX compliance and thus will need to address this requirement from scratch.

In summary, the one significant change anticipated for GenSAA to conform to the current Renaissance standards is in the area of POSIX compliance. In addition, future Renaissance requirements to follow development standards could necessitate additional changes to the GenSAA development approach.

4.3.1.3 User Interface Guidelines

The status of GenSAA compliance with user interface guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Import/Export	not Motif	Custom drag-and-drop implemented
Scripting	no req't	Prototyping interface to TSTOL
Customizable by user	yes	With workbench or TAE resource files
Decoupled from app	yes	Not as decoupled as original design
Layout/code separate?	yes, ASCII	Store layouts in GenSAA internal format
Custom widgets	not portable	Widgets developed by GenSAA are tied to special manager class.

GenSAA currently possesses requirements for user transfer of data between GenSAA elements though a drag-and-drop interface. This transfer was implemented prior to Motif 1.2 and was thus created in a custom fashion. GenSAA has no current requirement to support scripting. GenSAA currently allows extensive customization by system integrators and end users through its own workbench. Additional changes can be made by system integrators using the TAE workbench.

The GenSAA user interface component is only partially isolated from the data manger and expert system components. A complete segmentation of GenSAA into these three components would be more consistent with the Renaissance functional decomposition into user, application, and data services. This decoupling would
be a major change and would occur after the ACE timeframe. GenSAA display layouts are maintained in a custom ASCII format. GenSAA has created several custom widgets that currently only function when parented under a special GenSAA manager widget.

In summary, short term changes to GenSAA would be reworking their custom widgets so they are of general utility, and moving to a Motif standard approach for drag-and-drop. Long-term efforts should consider separating the user interface and application into separate processes and looking at UIL as a display definition storage mechanism. The near term costs in this area appear to be moderate in nature.

4.3.1.4 Renaissance Guidelines

The status of GenSAA compliance with Renaissance guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Platforms	HP, Sun	SGI by spring 1995?
Cost	TAE, CLIPS	TAE used in workbench, not run-time
Ren Interfaces	TBD	
Documentation	yes	All necessary information in User's Guide
Maturity	Rel 2	
On-line Help	partial	User's Guide in Mosaic format now
Performance	yes	Use working dialogs in few required places
Portability	been done	Completed Sun to HP port
Support Availability	yes	Support office in place
Verbatim reuse	yes	

GenSAA is currently supported on the HP and Sun platforms. A port to the SGI platforms scheduled for early 1995. GenSAA uses the CLIPS and TAE products developed by NASA (and thus available at no cost). Changes to support the Renaissance software backplane are currently undefined and could range from minor to major depending upon the similarity with current TPOCC data server communication mechanisms. In any case, GenSAA will need to add the capability to log system-level event messages. GenSAA possesses complete user documentation for both system integrators and end users in their User's Guide.

GenSAA has not yet implemented an online help facility. Current efforts in this area have centered on placing their user's guide into Mosaic format. If Mosaic is adopted, the issue of compatibility with the CDE help system currently planned for Renaissance would need to be resolved. GenSAA meets all performance, support, and reuse requirements listed in the implementation guide.

In summary, GenSAA will require several changes to meet the Renaissance guidelines that have been established. Significant software changes may be needed to support new Renaissance intertask communication mechanisms. A common online help approach for GenSAA and Renaissance could also be a key factor in the level of change required to meet these guidelines.

4.3.2 Action Plan

The GenSAA project currently is under the configuration control of the Code 510 CCB. CCRs may be submitted on GenSAA to allow interested parties to request a change and receive back a response from the GenSAA project. All changes to GenSAA identified in this transition plan will be turned into CCRs. Each CCR will be costed and resources to implement the change will be allocated if the CCR is approved. The changes will be undertaken according to a TBD schedule.

4.4 SpaceCAM

SpaceCam is a computer graphics system that enhances spacecraft operations and mission planning activities by giving analysts the ability to view, as if looking through a camera, a spacecraft as it orbits the Earth. SpaceCam features multiple views looking at, and from, the spacecraft that collectively provide an understanding of the position and orientation of the spacecraft relative to the Earth, Sun, Moon, stars, other spacecraft, and regions of magnetic interference. Provided as well are accurate fields of view from onboard instruments and antennas, the orientation of movable antennas and solar arrays as reported by telemetry, and the orbit path of the spacecraft. The high resolution color display has a "point and click", mouse-driven user interface that allows the user to easily change viewing points and control rendering. Objects in the display can be shown either as solid, translucent, or transparent (wire-frame outline) to get the best understanding of the spacecraft's situation.

The following subsections describe the approach for transitioning the SpaceCam for use with Renaissance. The first subsection compares the existing legacy building block with the user services standards and guidelines defined in the *Renaissance User Interface Implementation Guide*. The remaining subsections describe the actions to be taken to update SpaceCam and the schedule for completing these activities.

4.4.1 Mapping to Standards and Guidelines

This section presents the results of the comparison of the current SpaceCam implementation with the standards and guidelines presented in the *Renaissance User Interface Implementation Guide*. The results are divided up into the four categories of standards and guidelines defined in that document.

4.4.1.1 User Interface Standards

The status of SpaceCam compliance with user interface standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
X11	R5	
Motif	1.2.2	
Common Desktop Env	no	
Ren Style Principles	yes. so far	Strive for Motif Style Guide compliance

The SpaceCam software is based on X11R5 and Motif 1.2.2 -- versions of these commercial user interface technologies that are functionally equivalent to what has been chosen for the initial release of CDE. However, SpaceCam has not yet integrated with a desktop manager product such as CDE. SpaceCam will therefore need to build the icons, scripts, action definitions, and file types for CDE integration from scratch. SpaceCam has always worked toward *OSF/Motif Style Guide* compliance. It is anticipated that SpaceCam with thus have a high degree of compliance with the Renaissance style principles that will be based on this style guide.

In summary, the one significant transition to meet the user interface standards requirements is to achieve CDE compliance. Some changes to conform to Renaissance style principles may be required but are not anticipated to be major in nature.

4.4.1.2 Renaissance Standards

The status of SpaceCam compliance with Renaissance standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
Language	C++	Also FORTRAN legacy code in ephemeris library
POSIX	no	Not checked yet
Ren Develop Stds	TBD	Will convert to Imake as part of port to HP platform

All SpaceCam development is currently in C++, with FORTRAN used only in a legacy software library that support incorporation of ephemeris data. The SpaceCam project had no existing requirement for POSIX compliance and thus will need to address this requirement from scratch.

In summary, the one significant change anticipated for GenSAA to conform to the current Renaissance standards is in the area of POSIX compliance. In addition, future Renaissance requirements to follow development standards could necessitate additional changes to the GenSAA development approach.

4.4.1.3 User Interface Guidelines

The status of SpaceCam compliance with user interface guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Import/Export	partial	Publish results to TPOCC data server; screen snap to Postscript printer
Scripting	user views	Entire set of settings can be saved as a "user view" for later recall
Customizable by user	res files	Use standard X resource data base files
Decoupled from app	yes	
Layout/code separate?	yes, UIL	Inventor-supplied window frame layout not defined in UIL
Custom widgets	no	Use GenSAA widget wrapper; also TPOCC data source/server objects

SpaceCam does support data exchange with other applications by publishing some of its results to a copy of the TPOCC data server process. They also support screen snaps to a Postscript printer. However, no data import/export capability under user control is supported. SpaceCam has powerful features to support customization and user automation. Besides customization with X resource files, the SpaceCam user has the ability to tailor their three-dimensional views of the spacecraft environment and save these customizations to a user view file. SpaceCam also meets all guidelines for decoupling from the underlying application and maintenance of the screen layout independent of the code. SpaceCam has not created any custom widgets, though they have incorporated in the GenSAA widget wrapper software to simplify their use of TPOCC widgets for data service in C++.

In the long term, SpaceCam may need to improve their support for Motif data transfer techniques such as drag-and-drop or a clipboard. However, this appears to be the only change required to conform to the user interface guidelines.

4.4.1.4 Renaissance Guidelines

The status of SpaceCam compliance with Renaissance guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Platforms	SGI	Port to HP by March 1995; require hardware Z buffer
Cost	STK	Also OpenInventor and OpenGL (beta test site for HP versions)
Ren Interfaces	TBD	Use TPOCC data server library routines
Documentation	partial	User's Guide is a chapter in XTE's, need an implementation guide
Maturity	Prototype	Release to GOES December 1994, to XTE March 1995
On-line Help	no	
Performance	no	
Portability	untested	
Support Availability	partial	Developers currently set up SpaceCam with each mission
Verbatim reuse	yes	Also have several C++ objects of general utility to Renaissance

SpaceCam is currently supported only on the SGI platform. A port to the HP platform will be completed as soon as the OpenInventor and OpenGL products are made available on the HP. The current schedule anticipates that this operational release to the XTE MOC on the HP platform will occur in March 1995. An earlier SGI release may be made to the GOES project as early as December 1994. Note that for performance reasons SpaceCam does require display hardware that supports a hardware Z buffer.

Changes to support the Renaissance software backplane are currently undefined and could range from minor to moderate depending upon the similarity with current TPOCC data server communication mechanisms. SpaceCam will also need to plan for major changes to set up the documentation and personnel to support use of this software product by mission development teams acting as system integrators. This enhanced support will need to include online help for the users and an implementation guide for system integrators. Currently, involvement from the SpaceCam development team is required to set up this product for a new mission.

In summary, SpaceCam will require several significant changes to meet the Renaissance guidelines that have been established. At a minimum, online help, documentation for system integrators, and a support staff will need to be added.

4.4.2 Action Plan

The action plan and schedule for implementation of the Renaissance transition items for SpaceCam is TBD.

4.5 Generic Trend Analysis System (GTAS)

The Generic Trending and Analysis System, GTAS, provides a generic, user-friendly tool for assessing performance over the life of a spacecraft. There are six major features to the system: a telemetry and trending database, statistical calculations, mathematical manipulations of the data, graphical plots, relational telemetry expressions, and reports. GTAS provides the user easy access to the spacecraft data. It archives thirty days of telemetry, an extended duration of statistics, orbital information, and other miscellaneous data.

The following subsections describe the approach for transitioning GTAS for use with Renaissance. The first subsection compares the existing legacy building block with the user services standards and guidelines defined in the *Renaissance User Interface Implementation Guide.* The remaining subsections describe the actions to be taken to update GTAS and the schedule for completing these activities.

4.5.1 Mapping to Standards and Guidelines

This section presents the results of the comparison of the current GTAS implementation with the standards and guidelines presented in the *Renaissance User Interface Implementation Guide.* The results are divided up into the four categories of standards and guidelines defined in that document.

4.5.1.1 User Interface Standards

The status of GTAS compliance with user interface standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
X11	R5	
Motif	1.2.2	
Common Desktop Env	partial	Currently uses HP VUE (a principle CDE component)
Ren Style Principles	yes, so far	Compliant with the OSF/Motif Style Guide

The GTAS software is currently based on X11R5 and Motif 1.2. Thus, no problems are anticipated with these commercial libraries. GTAS has already performed integration with HP VUE. This previous work should give them a headstart when integrating with CDE. GTAS has always worked toward OSF/Motif Style Guide compliance. It is anticipated that GTAS with thus have a high degree of compatibility with the Renaissance style principles that will be based on this style guide.

In summary, the one notable transition for GTAS to meet the user interface standards requirements is to integrate with CDE. Some changes to conform to Renaissance style principles may be required but are not anticipated to be major in nature.

4.5.1.2 Renaissance Standards

The status of GTAS compliance with Renaissance standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
Language	C	Also SQL database interfaces
POSIX	partial	IPC compliant
Ren Develop Stds	TBD	

All GTAS development is currently in C, with no plans in place for migration to C++. GTAS has had a design goal of POSIX compliance but has not verified this fact to date.

In summary, the only change currently anticipated for GTAS to conform to the current Renaissance standards is a continuation of the effort to reach POSIX compliance. However, future Renaissance requirements to follow development standards or utilize object-oriented techniques could necessitate further changes to the GTAS development approach.

4.5.1.3 User Interface Guidelines

The status of GTAS compliance with user interface guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Import/Export	partial	Supports Postscript and archived data
Scripting	yes	Request file with editor
Customizable by user	yes	Also, speed button assignment
Decoupled from app	partial	User interface not decoupled from trend analysis application
Layout/code separate?	yes	UIL
Custom widgets	no	

GTAS currently possesses requirements for user transfer of data to a data base archive and Postscript printers. No drag-and-drop or clipboard interface is supported. GTAS currently allows customization by system integrators through the X Windows app-defaults file. Customization by the user is possible through the assignment of speed buttons.

The GTAS user interface component is not isolated from the trend analysis and data base components of this system. A complete segmentation of GTAS into these three components would be more consistent with the Renaissance functional decomposition into user, application, and data services. This decoupling would
be a major change and would occur after the ACE timeframe. GTAS display layouts are maintained in UIL format. GTAS has not created any custom widgets.

In summary, no substantial short term changes to GTAS are identified for this area. Long-term efforts should consider separating the user interface and application into separate processes and looking at drag-and-drop data interchange with other applications.

4.5.1.4 Renaissance Guidelines

The status of GTAS compliance with Renaissance guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Platforms	HP	HP/UX version 9.0
Cost	PV/Wave	PV/Wave used for plotting
Ren Interfaces	TBD	Currently supports TPOCC Telemetry Subset Files
Documentation	yes	
Maturity	Ver 1.08	In acceptance testing with frozen software
On-line Help	partial	Partial implementation
Performance	no	
Portability	not tested	No resources to support GTAS on multiple platforms
Support Availability	yes	
Verbatim reuse	yes	

GTAS is currently being developed on HP RISC processors running the HP/UX 9.0 operating system. No ports to other platforms have been attempted to date and work of this nature is beyond GTAS's current funding. PV/Wave is used for plotting and the ORACLE data base is used in Release 1 to store telemetry information. GTAS is currently investigating a move to a simple flat file data base instead of Oracle.

Changes required to GTAS interfaces because of the Renaissance software backplane are not expected to be major. GTAS already supports the TPOCC event logging interface. Furthermore, the primary interface with the rest of the MOC system is through telemetry subset files, not an interprocess communication mechanism.

GTAS already supports most TPOCC missions, so availability of support and documentation should not be a problem. Although online help has been partially implemented, more funding is needed to complete this work and to adopt the Renaissance help format. GTAS also will need to look at the performance feedback requirements for they are not fully compliant with this guideline.

In summary, GTAS will require several significant changes to meet the Renaissance guidelines that have been established. Online help, support for other reference platforms, and performance feedback will need to be added. Fortunately, the change to the new Renaissance intertask communication mechanisms should not have a significant effect on this building block.

4.5.2 Action Plan

The first release of GTAS was for support of the FAST mission. The current software development is to support all upcoming TPOCC missions. As the system is designed for generic trending and analysis, it will be easily ported to other missions and environments.

The current implementation release schedule is as follows:

August 1994	- Release 2	Interactive plots, TOSA integration, ingest/retrieval performance enhancements
March 1995	- Release 3	FDF/Formats interface, complete TOSA integration, enhanced reporting capabilities, performance enhancements

August 1995	- Release 4	Enhanced plotting capabilities, performance enhancements

As state above, it has been determined that some Renaissance SMEX/ACE funding will be necessary to provide a fully Renaissance compliant GTAS system.

4.6 MISSION OPERATIONS PLANNING AND SCHEDULING SOFTWARE (MOPSS) GRAPHICAL USER INTERFACE

The Mission Operations Planning and Scheduling Software (MOPSS) is a multi-user system utilized by both scientists and flight operations teams to concurrently plan and schedule spacecraft and ground system activities and resources. A powerful graphical user interface enables a user to view and manipulate schedule data. The adaptable architecture of MOPSS enables easy integration of tools and models to meet new system requirements. Data can be directly ingested from models or from the user interface, or indirectly from flat files in the format produced by existing systems.

The heart of the system is a flexible schedule manager component based on the Oracle 7 RDBMS. This schedule manager supports viewing and editing of schedule data concurrently by multiple users. Multiple independent schedules can be managed simultaneously. External processes can be invoked directly from the graphical display, or based on time. User privileges and authorization prevent breaching of data security and integrity. User-defined constraint checks ensure that generated schedules do not violate resource limitations.

The graphical user interface provides the user with visual access to any data maintained by the schedule manager. A choice of user-configurable display formats, such as timeline, calendar, and tabular representations, are envisioned. With the timeline representation, the user may visualize an entire schedule or any portion thereof. The user may navigate through the schedule using the zooming and scrolling features. The tabular display presents the schedule data in a textual form. The calendar representation displays planning data in a familiar wall-calendar format.

The following subsections describe the approach for transitioning the MOPSS graphical user interface to Renaissance compliance. The first subsection compares the MOPSS building block as currently planned with the user services standards and guidelines defined in the *Renaissance User Interface Implementation Guide.* The remaining subsections describe the actions to be taken to update the MOPSS graphical user interface and the schedule for completing these activities.

4.6.1 Mapping to Standards and Guidelines

This section presents the results of the comparison of the MOPSS graphical user interface implementation with the standards and guidelines presented in the *Renaissance User Interface Implementation Guide.* The results are divided up into the four categories of standards and guidelines defined in that document.

4.6.1.1 User Interface Standards

The status of the MOPSS user interface compliance with user interface standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
X11	R5	
Motif	1.2.2	Transition to 1.2.3 with preliminary CDE integration
Common Desktop Env	partial	Integrated with HP VUE (a principle CDE component)
Ren Style Principles	TBD	Style principles are still TBD

MOPSS is currently in development of Release 1 for the XTE and TRMM missions. MOPSS has inherited a strong legacy from TPOCC. Since MOPSS is currently planned as a subsystem of the MOC, MOPSS will continue to maintain compatibility with TPOCC.

Motif 1.2.3 has been chosen as the initial version for CDE. MOPSS will upgrade to 1.2.3 and CDE during the same release. MOPSS is currently at X11R5. MOPSS is currently compliant with the *OSF/Motif Style Guide*.

In summary, the one notable transition for MOPSS to meet the user interface standards requirements is to integrate with CDE. Some changes to conform to Renaissance style principles may be required but are not anticipated to be major in nature.

4.6.1.2 Renaissance Standards

The status of the MOPSS user interface compliance with Renaissance standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
Language	C, C++	Approximately 12k lines of inherited code are currently in C
POSIX	partial	System calls currently not supported by POSIX 1003.4 are utilized
Ren Develop Stds	TBD	Development standards are still TBD

MOPSS has inherited a significant amount of legacy code from the User Planning System (UPS). The UPS legacy code performs the low-level graphical timeline manipulation, and interfaces with X11 and Motif. This portion of MOPSS, known as the Graphical Timeline Viewer, will be maintained in C. All new code implemented in the Graphical Timeline Viewer will also be implemented in C.

Four higher-level segments make up the Graphical Timeline user interface. These include the Session Manager, Display Manager, Event Manager and Graphical Timeline Viewer Manager. These segments have been partitioned into objects and implemented in C++. The C/C++ interface has been isolated between the Graphical Timeline Viewer and the Graphical Timeline Viewer Manager. All new code implemented in these segments will continue to be in C++.

In summary, the one significant change anticipated for MOPSS to conform to the current Renaissance standards is in the area of POSIX compliance. In addition, future Renaissance requirements to follow development standards could necessitate additional changes to the MOPSS development approach.

4.6.1.3 User Interface Guidelines

The status of the MOPSS user interface compliance with user interface guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Import/Export	yes	where appropriate
Scripting	yes	Standard Schedule Representation File syntax
Customizable by user	yes	where appropriate
Decoupled from app	yes	database client communicates with server via SQL*Net
Layout/code separate?	yes	UIL/Resource files
Custom widgets	yes	3 currently developed, with a potential of approximately 10

MOPSS supports import/export of data from/to other user interface elements through a Motif-approved transfer technique where appropriate. "Drag and drop" are used extensively throughout

the graphical timeline and configuration panels. Printing of the timeline to a Postscript laser printer is also supported.

MOPSS provides a user scripting capability known as the Standard Schedule Representation File. The syntax of the language provides the user with the capability to duplicate any action that can be performed from the timeline. For enhanced capability, the user has the ability to invoke UNIX-level system scripts.

Because MOPSS is a generic mission planning and scheduling system, the user has the ability to define as many graphical timeline display configurations as are necessary. Display configuration can be either PUBLIC or private. They are created on-the-fly using the Display Configuration Editor, which is available as a pull-down from the MOPSS main panel.

The Graphical Timeline Viewer user interface building block is decoupled from that of the Data Manager application building block. A strictly controlled set of access routines are maintained to provide the interface to the Data Manager application. The Data Manager application is implemented using Oracle Relational Data Base Management Software (RDBMS). Access between the access routines and Oracle is via SQL*Net.

The layouts for the pull-down menus are maintained using Builder Xcessory. Builder Xcessory stores the resulting menu structure in UIL. This allows the user interface layouts for the pull-down menus to be maintained independently of the underlying user interface software.

In summary, MOPSS will need to contribute its widgets to the Renaissance widget library. However, this appears to be the only change required to conform to the user interface guidelines.

4.6.1.4 Renaissance Guidelines

The status of the MOPSS user interface compliance with Renaissance guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Platforms	HP, DEC	HP 9000/715
Cost	yes	Oracle RDBMS used for storing schedule data
Ren Interfaces	TBD	TBD
Documentation	Release 1	MOPSS System Requirements Specification (SRS)
	Release 1	MOPSS Capabilities Document
	Release 1	MOPSS Users Guide
	Release 3	MOPSS Implementation Guide
Maturity	Prototype	First release scheduled for December 1994
On-line Help	planned	Utilize CDE help facility
Performance	planned	Planned for MOPSS Release 3
Portability	<1 month	Ports easily to DEC
Support Availability	yes	MOPSS Working Group supports MOPSS (see below)
Verbatim reuse	yes	No coding changes required.

The MOPSS user interface software is currently being developed for an HP 9000/715/80 with 64Mb, or more, of memory and running under HP/UX 9.0. The user interface software was originally developed on a DEC Workstation 5000. Since being originally ported from the DEC, the software has been successfully ported back to the DEC for another application. Although MOPSS is currently in development for Release 1, a large component of the user interface software was reused from UPS.

MOPSS is a generic mission planning and scheduling system. In order to support multiple missions, MOPSS requires no coding changes to store new kinds of scheduled events. MOPSS is currently configured to accept PSAT, maneuver and UAV data from FDF, USM and SDN data from UPS, and FOT planning data from the MOC off-line. MOPSS can export scheduled data through the Standard Schedule Representation File or into an Activity Plan. New scheduling engines can be easily adapted to MOPSS through the Standard Schedule Representation File or interface directly with the database using a standard set of access procedures. Where necessary, mission specific models can be written as Oracle PL/SQL package procedures. Access to the database is strictly maintained through a well-defined set of access procedures.

In summary, MOPSS will require several moderate-level enhancements to meet the Renaissance guidelines that have been established. The addition of an online help capability and performance feedback changes are planned but not until 1996. Effort will also be required to move to the Renaissance software backplane mechanisms but the severity of such a change can not be determined at this time.

4.6.2. Action Plan

MOPSS has an existing mechanism to handle requests for updates to MOPSS technology. The MOPSS Working Group (MOPSSWG) was established by GSFC Code 510 and its SEAS contractors to ensure technical continuity, communications, and proper integration of MOPSS components. It is the responsibility of the MOPSSWG to oversee MOPSS development, and evaluate new requirements, modifications, changes, DRs, etc. The MOPSSWG will be the controlling board for MOPSS documentation, reporting to the MOD CCB. This process is documented in the *MOPSS Project Support Plan*.

MOPSS is currently scheduled to deliver major software releases in support of the XTE and TRMM MOCs. Additional requirements are currently being developed to support the ACE MOC.

MOPSS Release 1 will be incorporated into XTE Release 2, then six months later into TRMM Release 1. MOPSS Release 2 will be incorporated into XTE Release 3 and TRMM Release 2. MOPSS Release 3 is currently envisioned to support TRMM Release 3, and will eventually be retrofitted into the XTE MOC after launch of the XTE spacecraft. Releases which support the ACE mission have not been planned at this time, however it is anticipated that MOPSS Release 3 will also support ACE MOC Release 1. MOPSS Release 4 has been planned to support the ACE MOC Launch Release (Release 2) delivery. A fifth release of MOPSS is planned for ACE in the first quarter or 1997.

MOPSS Release 1	December 1994
MOPSS Release 2	April 1995
MOPSS Release 3	March 1996
MOPSS Release 4	October 1996
MOPSS Release 5	March 1997

These are the dates the MOPSS software is officially delivered; in most cases, the software is available for integration by the MOCs two months earlier.

Capabilities for implementation have been allocated through MOPSS Release 2. Requirements for the TRMM MOC have been defined for MOPSS Release 3, but not yet for ACE. Renaissance requirements for ACE would need to be contained in Release 3 or later.

MOPSS Release 3:	On-line help facility POSIX compliance Basic CDE integration Motif 1.2.3 User feedback
MOPSS Release 4:	Full CDE integration

4.7 User Interface and Executive (UIX)

The User Interface and Executive (UIX) is a configurable user interface designed to support applications executing in the Unix workstation environment within the Flight DynamicsDistributed System (FDDS). UIX gives FORTRAN, C and Ada developers the ability to create an X Window GUI without requiring X programming knowledge or the use of a GUI builder. A secondary goal of UIX is to reduce training costs for operations personnel by standardizing the types of displays in use.

The following subsections describe the approach for transitioning UIX for use with Renaissance. The first subsection compares the existing legacy building block with the user services standards and guidelines defined in the *Renaissance User Interface Implementation Guide.* The remaining subsections describe the actions to be taken to update UIX and the schedule for completing these activities.

4.7.1 Mapping to Standards and Guidelines

This section presents the results of the comparison of the current UIX implementation with the standards and guidelines presented in the *Renaissance User Interface Implementation Guide.* The results are divided up into the four categories of standards and guidelines defined in that document.

4.7.1.1 User Interface Standards

The status of UIX compliance with user interface standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
X11	R4	Moving to X11R5 by January 1995
Motif	1.1	Moving to Motif 1.2 by January 1995
Common Desktop Env	partial	Already integrated with SCO desktop environment
Ren Style Principles	yes, so far	Motif compliant

The UIX software is currently based on X11R4 and Motif 1.1. The existing UIX development schedule calls for an upgrade by January 1995 to X11R5 and Motif 1.2 -- the same versions of these commercial user interface technologies that have been chosen for the initial release of CDE. Also, UIX has already performed
integration with the desktop manager included with SCO UNIX. This previous work should give them a headstart when integrating with CDE. UIX has always worked toward OSF/Motif Style Guide compliance. It is anticipated that UIX with thus have a high degree of compatibility with the Renaissance style principles that
will be based on this style guide.

In summary, the one notable transition for UIX to meet the user interface standards requirements is to integrate with CDE. Some changes to conform to Renaissance style principles may be required but are not anticipated to be major in nature.

4.7.1.2 Renaissance Standards

The status of UIX compliance with Renaissance standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
Language	C	Object-oriented design
POSIX	full	Completely compliant
Ren Develop Stds	TBD	

All UIX development is currently in C, though the system is designed and implemented using object-oriented techniques. Future migration to C++ will be considered in the Flight Dynamics Division (FDD) selects this language as the basis for future implementations. UIX does have an existing requirement for POSIX compliance and has been able to meet this requirement in all cases to date.

In summary, no significant changes are currently anticipated for UIX to conform to the current Renaissance standards. However, future Renaissance requirements to follow development standards or utilize object-oriented techniques could necessitate changes to the UIX development approach.

4.7.1.3 User Interface Guidelines

The status of UIX compliance with user interface guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Import/Export	planned	Plan to use Motif clipboard for text transfers
Scripting	session mgr	Session manager allows sequencing at the application level
Customizable by user	partial	Through X resource data base; long-term plan is for GUI interface
Decoupled from app	partial	Separated into different libraries but run as part of same process
Layout/code separate?	yes, ASCII	Store layouts in UIX internal format
Custom widgets	none	May want to utilize X-Y plot in Renaissance library

UIX currently possesses requirements for user transfer of data with other applications through a cut and paste to a clipboard. This technique is a Motif-approved method, though a drag-and-drop interface will be evaluated after
UIX migrates to Motif 1.2. UIX supports sequencing of programs with the session manager but can not currently support sequences of user directives to a single application. UIX currently allows extensive customization by system integrators through the X Windows app-defaults file and UIX display definitions. A graphical interface for user customization is planned for the future.

The UIX user interface software is isolated from the underlying application in separate libraries but does not run as a standalone process. A design using remote procedure calls (RPCs) to further isolate these functions in separate processes was considered over a year ago. It was rejected at the time due to performance concerns and the lack of ADA or FORTRAN bindings for the RPC calls. Such an approach will be examined again to see if these concerns can be alleviated and the redesign achieved at moderate cost. This segmentation would be more consistent with the approaches put forward in the current definition of the Renaissance software backplane.

UIX display layouts are maintained in a custom ASCII format. UIX has not created any custom widgets though the project does use the XbaeMatrix widget from the public domain and is interested in evaluating the TPOCC X-Y plot widget to implement some specialized plotting requirements.

In summary, UIX should strongly consider separating the user interface and application into separate processes in the near term. Longer term work could focus on drag-and-drop capabilities, scripting of individual actions within a program, and looking at UIL as a display definition storage mechanism. The near term costs in this area appear to be moderate in nature.

4.7.1.4 Renaissance Guidelines

The status of UIX compliance with Renaissance guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Platforms	Sun,SCO,HP	SCO currently: Sun and HP by January 1995
COTS Usage	CodeBase	Have CodeBase source and also use FDF NCSS library
Ren Interfaces	TBD	
Documentation	yes	Programmer's Reference document is planned
Maturity	Prototype	Release 1 by January 1995
On-line Help	Mosaic?	Investigating Mosaic as an on-line help package
Performance	yes	Working dialogs included
Portability	tested	Prototype was ported to several platforms
Support Availability	partial	Have not planned for support outside of Code 550
Verbatim reuse	yes	

UIX is currently supported on the SCO UNIX platform. A port to the SGI platforms was completed with a UIX prototype; the Sun and HP platforms will both be fully supported by early 1995. UIX uses the CodeBase API to provide relational data base management; however, the FDD Center of Expertise (COE) has the source
code for this product and there is no embedded cost for this product. Changes to support the Renaissance software backplane are currently undefined and could range from minor to major depending upon the similarity with current UIX communication mechanisms. Also note that UIX plans a programmer's reference document (a preliminary version is available at this time).

UIX hopes to support an online help capability by early 1995. Current efforts in this area center on investigating the hypertext and editing capabilities of Mosaic. Mosaic is an attractive alternative since it meets the FDD requirement for all documents to be maintained and distributed in electronic format. If Mosaic is adopted, the issue of compatibility with the CDE help system currently planned for Renaissance would need to be resolved. UIX meets all performance and reuse requirements listed in the implementation guide. Although support for the use of UIX is being provided to Code 550 projects, support of outside users has not been considered prior to this time.

In summary, UIX will require several changes to meet the Renaissance guidelines that have been established. A support staff for users outside the FDD will need to be added and significant software changes may be needed to support new Renaissance intertask communication mechanisms. A common online help approach for UIX and Renaissance could also be a key factor in the level of change required to meet these guidelines.

4.7.2 Action Plan

The UIX project has an existing software change request (SCR) mechanism to handle requests for updates to UIX capabilities. This SCR mechanism allows interested parties to request a change and receive back a response from the UIX project. All changes to UIX identified in this transition plan will be turned into SCRs. Each SCR will be costed and resources to implement the change

will be allocated if the SCR is approved. The changes will be undertaken according to a TBD schedule.

4.8 PACOR II Information and Control Subsystem (PICS)

The PACOR II Information and Control Subsystem (PICS) is responsible for providing the computer-human interface for control, monitoring, and information retrieval for the PACOR II software, Major functions of the PICS include acquisition scheduling, resource allocation, system monitoring, data retention, and reporting.

The following subsections describe the approach for transitioning PICS for use with Renaissance. The first subsection compares the existing legacy building block with the user services standards and guidelines defined in the *Renaissance User Interface Implementation Guide.* The remaining subsections describe the actions to be taken to update PICS and the schedule for completing these activities.

4.8.1 Mapping to Standards and Guidelines

This section presents the results of the comparison of the current PICS implementation with the standards and guidelines presented in the *Renaissance User Interface Implementation Guide.* The results are divided up into the four categories of standards and guidelines defined in that document.

4.8.1.1 User Interface Standards

The status of PICS compliance with user interface standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
X11	R5	
Motif	1.2.3	
Common Desktop Env	no	No file manager or desktop integrated to date
Ren Style Principles	yes, so far	Strive for compliance with Motif Style Guide and PACOR II guidelines

The PICS CHI software is based on X11R5 and Motif 1.2.3 -- the same versions of these commercial user interface technology that have been chosen for the initial release of CDE. However, the PACOR II user environment has not yet included a desktop manager product such as CDE. PICS will therefore need to build the icons, scripts, action definitions, and file types for CDE integration from scratch. PACOR II has always worked toward *OSF/Motif Style Guide* compliance and has followed an internal set of style principles. It is anticipated that PICS with thus have a high degree of compliance with the Renaissance style principles
that will be based on such principles.

In summary, the one significant transition to meet the user interface standards requirements is to achieve CDE compliance. Some changes to conform to Renaissance style principles may be required but are not anticipated to be major in nature.

4.8.1.2 Renaissance Standards

The status of PICS compliance with Renaissance standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
Language	C	Limited experience with object-oriented technologies
POSIX	isolated	BSD socket usage isolated in library; otherwise, POSIX compliant
Ren Develop Stds	TBD	Development standards are still TBD

All PICS development is currently in C. No plans have been made to consider an object-oriented programming environment for the future. If Renaissance requires such an approach, PACOR personnel would require training and lead time to convert to the new methodology. PACOR II does have an existing requirement for
POSIX compliance. All system calls that are not defined within the POSIX specifications must be isolated in separate libraries. This isolation has been achieved by the Berkeley socket library calls used in PICS.

In summary, no significant changes are currently anticipated for PICS to conform to the current Renaissance standards. However, future Renaissance requirements to follow development standards or utilize object-oriented techniques could necessitate substantial change to the PICS development approach.

4.8.1.3 User Interface Guidelines

The status of PICS compliance with user interface guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Import/Export	no req't	
Scripting	no req't	
Customizable by user	partial	Only supported through X windows app-defaults files
Decoupled from app	yes	Socket interface for control; SQL interface to get data
Layout/code separate?	yes, UIL	Builder Xcessory used to maintain UIL
Custom widgets	7	Half of the widgets are standalone already; all have man pages written

PICS currently possesses no requirements for user transfer of data with other applications or scripting of user operations. Data exchange with other processes is accomplished through SQL calls to the Oracle data base. PICS
currently allows customization through the X Windows app-defaults file in the user's home directory, although this capability is not currently heavily used. No menu or graphical interface for user customization is currently supported. The PICS user interface software is clearly decoupled from the underlying application and the layout is also maintained separately as UIL. PICS uses the WML facility to augment the UIL compiler with information on their custom widgets. Half of the 7 existing PICS widgets are close to ready to be contributed to a Renaissance widget library; the others contain PACOR library calls that would have to be removed. PICS also uses 2 widgets from Bellcore obtained from the Internet.

In summary, PICS will need to contribute its widgets to the Renaissance widget library and should add additional customization options to meet the long-term Renaissance goal of a high degree of user customization. However, none of these changes to conform to user interface guidelines will cause major impacts in the near term.

4.8.1.4 Renaissance Guidelines

The status of PICS compliance with Renaissance guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Platforms	Solaris	Solaris 2.3
COTS Usage	Oracle	CHI contains embedded SQL in an isolated library
Ren Interfaces	TBD	
Documentation	partial	No current plan for documentation for system integrators
Maturity	Rel 1	
On-line Help	planned	Being given low priority so far
Performance	partial	Adding user feedback of this type; only needed in a few situations
Portability	started	Investigating port to HP as initial Renaissance testbed activity
Support Availability	limited	No plans to support system integration teams
Verbatim reuse	yes	

PICS is currently being developed on Sun SPARC processors running the Solaris 2.3 operating system. No ports to other platforms have been attempted to date, though a Renaissance testbed activity has been initiated to begin studying the impacts caused by such a port. SQL calls are used to retrieve data for the PICS display; these Oracle routines are the only COTS package being utilized at run-time by PICS.

Changes to PICS interfaces with other processes are likely with Renaissance and will cause a significant software impact. The new common event format and event logging call will require PICS software changes to adapt to the new calling sequence and message formatting requirements. If process control is no longer through sockets, the impact will be even larger.

PICS will also need to plan for major changes to set up the documentation and personnel to support use of this software product by mission development teams acting as system integrators. This enhanced support will need to include online help for the users. Although online help is a current PICS requirement, it has not yet been given a high priority and is not scheduled for implementation in the near term.

In summary, PICS will require several significant changes to meet the Renaissance guidelines that have been established. Online help, documentation for system integrators, and a support staff will need to be added. Software porting efforts will need to be stepped up, and significant software changes may be needed to support new Renaissance intertask communication mechanisms.

4.8.2 Action Plan

The PACOR II project has an existing mechanism to handle requests for updates to PACOR II technology. This DCR mechanism allows interested parties to request a change and receive back a response from the PACOR II Configuration Management Board All changes to PICS identified in this transition plan will be turned into PACOR II DCRs. Each DCR will be costed and resources to implement the change will be allocated if the DCR is approved. The changes will be undertaken according to a TBD schedule.

4.9 Quality Analysis Workstation Subsystem (QAWS)

The Quality Analysis Workstation Subsystem (QAWS) serves as an off-line analysis tool server. The system is designed to permit troubleshooting of data/spacecraft, internal system, and network anomalies without impact to ongoing operations.

The following subsections describe the approach for transitioning QAWS for use with Renaissance. The first subsection compares the existing legacy building block with the user services standards and guidelines defined in the *Renaissance User Interface Implementation Guide*. The remaining subsections describe the actions to be taken to update QAWS and the schedule for completing these activities.

4.9.1 Mapping to Standards and Guidelines

This section presents the results of the comparison of the current QAWS implementation with the standards and guidelines presented in the *Renaissance User Interface Implementation Guide*. The results are divided up into the four categories of standards and guidelines defined in that document.

4.9.1.1 User Interface Standards

The status of QAWS compliance with user interface standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
X11	R5	
Motif	1.2.3	
Common Desktop Env	no	No file manager or desktop integrated to date
Ren Style Principles	yes, so far	Strive for compliance with Motif Style Guide and PACOR II guidelines

The QAWS CHI software is based on X11R5 and Motif 1.2.3 -- the same versions of these commercial user interface technology that have been chosen for the initial release of CDE. However, the PACOR II user environment has not yet included a desktop manager product such as CDE. QAWS will therefore need to build the icons, scripts, action definitions, and file types for CDE integration from scratch. PACOR II has always worked toward *OSF/Motif Style Guide* compliance and has followed an internal set of style principles. It is anticipated that QAWS with thus have a high degree of compliance with the Renaissance style principles
that will be based on such principles.

In summary, the one significant transition to meet the user interface standards requirements is to achieve CDE compliance. Some changes to conform to Renaissance style principles may be required but are not anticipated to be major in nature.

4.9.1.2 Renaissance Standards

The status of QAWS compliance with Renaissance standards is shown in the table below. A detailed discussion follows the table.

STANDARD	STATUS	COMMENTS
Language	C	Limited experience with object-oriented technologies
POSIX	isolated	BSD socket usage isolated in library; otherwise, POSIX compliant
Ren Develop Stds	TBD	Development standards are still TBD

All QAWS development is currently in C. No plans have been made to consider an object-oriented programming environment for the future. If Renaissance requires such an approach, PACOR personnel would require training and lead time to convert to the new methodology. PACOR II does have an existing requirement for
POSIX compliance. All system calls that are not defined within the POSIX specifications must be isolated in separate libraries. This isolation has been achieved by the Berkeley socket library calls used in QAWS.

In summary, no significant changes are currently anticipated for QAWS to conform to the current Renaissance standards. However, future Renaissance requirements to follow development standards or utilize object-oriented techniques could necessitate substantial change to the QAWS development approach.

4.9.1.3 User Interface Guidelines

The status of QAWS compliance with user interface guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Import/Export	no req't	
Scripting	no req't	
Customizable by user	partial	Only supported through X windows app-defaults files
Decoupled from app	yes	Socket interface for control; SQL interface to get data
Layout/code separate?	yes, UIL	Builder Xcessory used to maintain UIL
Custom widgets	1	Widget man page written

QAWS currently possesses no requirements for user transfer of data with other applications or scripting of user operations. Data exchange with other processes is accomplished through SQL calls to the Oracle data base. QAWS currently allows customization through the X Windows app-defaults file in the user's home directory, although this capability is not currently heavily used. No menu or graphical interface for user customization is currently supported. The QAWS user interface software is clearly decoupled from the underlying application and the layout is also maintained separately as UIL. However, the application portions of QAWS should eventually be moved out of the user services functional area. The QAWS widget will only need minor changes to be readied for submission to the Renaissance widget library.

In summary, QAWS will need to contribute its widget to the Renaissance widget library in the short-term. Long-term changes include adding additional customization options to meet the Renaissance goal of a high degree of user customization and moving some software elements into other Renaissance functional areas. However, none of these changes to conform to user interface guidelines will cause major impacts in the near term.

4.9.1.4 Renaissance Guidelines

The status of QAWS compliance with Renaissance guidelines is shown in the table below. A detailed discussion follows the table.

GUIDELINE	STATUS	COMMENTS
Platforms	Solaris	Solaris 2.3
COTS Usage	Oracle	CHI contains embedded SQL in an isolated library
Ren Interfaces	TBD	
Documentation	partial	No current plan for documentation for system integrators
Maturity	Rel 1	
On-line Help	planned	Being given low priority so far
Performance	partial	Adding user feedback of this type; only needed in a few situations
Portability	started	Investigating port to HP as initial Renaissance testbed activity
Support Availability	limited	No plans to support system integration teams
Verbatim reuse	yes	

QAWS is currently being developed on Sun SPARC processors running the Solaris 2.3 operating system. No ports to other platforms have been attempted to date, though a Renaissance testbed activity has been initiated to begin studying the impacts caused by such a port. SQL calls are used to retrieve data for the QAWS display; these Oracle routines are the only COTS package being utilized at run-time by QAWS.

Changes to QAWS interfaces with other processes are likely with Renaissance and will cause a significant software impact. The new common event format and event logging call will require QAWS software changes to adapt to the new calling sequence and message formatting requirements. If process control is no longer through sockets, the impact will be even larger. Furthermore, the QAWS data interface will also need to be changed to reflect the new environment in which this tool will be used.

QAWS will also need to plan for major changes to set up the documentation and personnel to support use of this software product by mission development teams acting as system integrators. This enhanced support will need to include online help for the users. Although online help is a current QAWS requirement, it has not yet been given a high priority and is not scheduled for implementation in the near term.

In summary, QAWS will require several significant changes to meet the Renaissance guidelines that have been established. Online help, documentation for system integrators, and a support staff will need to be added. Software porting efforts will need to be stepped up, and significant software changes will be needed to support new Renaissance intertask communication mechanisms.

4.9.2 Action Plan

The PACOR II project has an existing mechanism to handle requests for updates to PACOR II technology. This DCR mechanism allows interested parties to request a change and receive back a response from the PACOR II Configuration Management Board All changes to QAWS identified in this transition plan will be turned into PACOR II DCRs. Each DCR will be costed and resources to implement the change will be allocated if the DCR is approved. The changes will be undertaken according to a TBD schedule.

4.10 SOHO Simulator USER INTERFACE

Information on the transition of the SOHO simulator user interface is TBS.